Implementation of a column interleaving function with a limited amount of columns

Field of the invention

[0001] The present invention is related to an implementation of an interleaver, used in communication systems to protect the transmitted data.

State of the art

[0002] In communication systems interleaving is an often-used technique, mostly in conjunction with coding techniques, to protect data signals against burst errors.

[0003] In a column-interleaving function, data entities are written into rows and after column permutation, read from columns. For a bit interleaver, the data entity is a logical one or zero.

[0004] Interleaving functions require some kind of memory structure. The technical problem addressed by this invention is that of implementing said memory structure in a more efficient way.

Aims of the invention

[0005] The present invention aims to provide an efficient memory structure required in the implementation of a column interleaving function that overcomes the disadvantages of the prior art solutions.

Summary of the invention

[0006] The invention relates to a method to implement a column interleaving function, comprising the steps of:

- Providing a number of memories equal to the maximum number of columns in the interleaver function,

- Inputting a stream of data entities,
- Writing said data entities successively into a memory, until all memories are completely filled or until all data entities are written,
- Performing selection and permutation on said memories,
- Reading out said data entities in said permuted memories, in a memory-by-memory fashion.

[0007] In an advantageous embodiment the data entities in the input stream are first written into a register and when the register is filled, the step of writing into a memory is applied.

[0008] In one embodiment of the invention the data entities are logical ones and zeros. In another embodiment the data entities are multiple bit words. In a specific embodiment the data entities are three bit words.

[0009] Preferably the register is arranged to store each multiple bit word at one location in said memories.

[0010] In a specific embodiment the number of columns used in the column interleaver function is changed on the fly, whereby the number of columns does not exceed the maximum number of columns.

[0011] As a second object the invention relates to a module for column interleaving, comprising means for applying a method as previously described.

[0012] In another preferred embodiment an integrated circuit device comprises such a module.

[0013] Typically a communication system device comprises a module or an integrated circuit device as described above.

[0014] In an advantageous embodiment a spread-spectrum communication apparatus comprises such a module or an integrated circuit device.

[0015] As another object the invention relates to a column interleaver, comprising a number of memories equal to the maximum number of columns desired in the interleaver and means to perform column selection and permutation. In a specific embodiment the column interleaver further comprises a register.

Short description of the drawings

[0016] Fig. 1 represents the concept of a column interleaving function.

[0017] Fig. 2 represents a prior art solution.

[0018] Fig. 3 represents a solution according to the invention.

[0019] Fig. 4 represents an embodiment of the invention with three bit words, still performing bit-based interleaving.

Detailed description of the invention

[0020] Figure 1 shows the concept of a column interleaving function. Data entities are written into a memory on a row-by-row basis. After permuting the columns they are read from the columns. In a bit interleaver the data entities are bits, i.e. logical ones and zeros.

[0021] In some application areas, the amount of rows and columns are flexible, depending of the data block size. The invention described here can have a flexible amount of rows but requires a limited amount of columns.

[0022] Figure 2 shows the prior art solution. Serial data is written into a bit RAM on consecutive addresses. This writing represents the interleaver row-writing action. A bit RAM is used because column selection and permutation is bit-based. This column selection and permutation is implemented by the address calculation for reading out the

data. With this solution there are several disadvantages like relatively high power consumption and memory requirements.

[0023] By using a smart memory structure and by grouping the input data before writing to memory, the interleaver implementation can be improved in several ways:

- Less power consumption by reducing the amount of memory accesses and by using distributed memory,
- Increased memory read-write bandwidth by removing the serial bottleneck and by changing the data entities from bits to multiple bit words,
- Very straightforward address calculation.

[0024] Figure 3 discloses an implementation according to the invention. In case of a serial data input stream data is first written into a shift register. Also parallel data not having the right format (number of parallel streams) is first written into a shift register. The width of this register equals the amount of columns of the interleaver function. Thus, if the number of parallel data streams corresponds to the number of columns, no input register is required. Once the register is filled completely, each bit is written into a separate bit-oriented memory. There is one memory for every column defined in the interleaver function. When the column data is read out, the data is already sorted into columns. One only has to select the correct memory to perform the permutation function (done by multiplexer). While reading out the columns, there is only one memory active at the time.

[0025] As mentioned before, the input register is not really necessary for the interleaver function as such; it just synchronises the memory accesses and takes away possible data throughput bottlenecks. This latter feature

is very important. If, for example, the data is coming from a turbo encoder (this is the case in 3GPP outer modem), the encoder output generates multiple bits every clock cycle. Due to the use of multiple memories, one can write more than one bit to memory in one clock cycle and avoid possible wait states in the encoder data path or avoid the need to clock the memories on a higher frequency to prevent these wait states.

[0026] Figure 4 discloses in a specific embodiment of the invention an extra enhancement of the input register. By adapting the input register, one can use multiple bit words in the memories (instead of using bit-oriented memories) and still perform a bit-oriented permutation. As a result, the amount of memory accesses can be limited in addition to the decrease achieved by using multiple memories. This decrease improves the power consumption and increases the read/write bandwidth of the memories (with same clock). In Figure 4 a scheme employing three-bit words is shown. Schemes with other word lengths can be derived in a straightforward way.